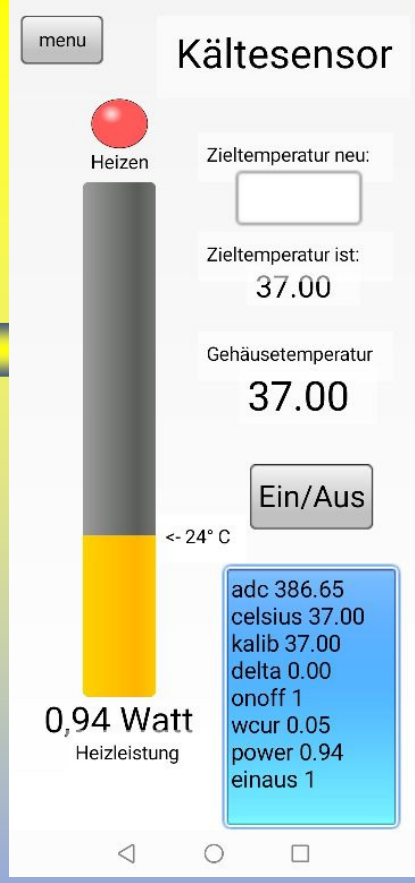
Kältesensor

Können wir unser Kälteempfinden messen?

Läßt sich "Frieren" bei Wind und Wetter messen?

Ja, mit einem speziellen Kältesensor! Ein neuer Ansatz kann wichtige Fragen beantworten.



Motivation

- Manchmal friert man im Winter - und manchmal nicht. Unser Kälteempfinden ist nicht in Grad Cesius angebbar. Luftfeuchte, Regen oder Windstärke mischen mit.
- Um einen Kälte-Empfindungs-Sensor zu bauen, braucht man mindestens ein Thermometer, einen Feuchtemesser, einen Regenmesser und einen Windmesser. Und dann noch hochkomplizierte Vorhersagemodelle, für die man keine Referenzwerte zur Validierung findet. Pure Empirik?
- Was also tun? Ganz einfach: Man nehme einen Körper, der grob das Größenverhältnis unseres Körpers hat und heize ihn auf 37°C auf.
- Die dazu nötige Heizleistung ist dann ein Maß unseres Kälteempfindens.
- Als Stubenhocker lieben wir eine Zimmertemperatur von mindestens 24°C - damit haben wir einen Referenzpunkt für die "Normalleistung"
- Nun können wir den Sensor nach draußen in die Kälte hängen und sehen, was geschieht. Wir können ihm Mützen und Pullover anziehen, um die Heizleistung wieder auf "Normalleistung" herunter zu bringen.
- Ein Thermostat hält die Alu-Hülle des Sensors konstant auf der eingestellten Zieltemperatur. Die dafür benötigte (Wärme-) Leistung wird gemessen und an Bluetooth ausgegeben.

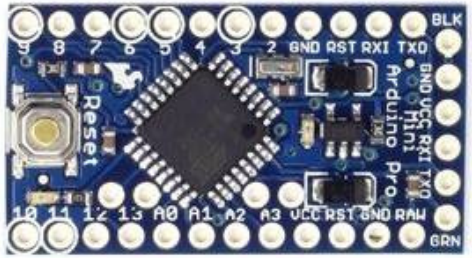
Funktion

- Der Sensor hängt an geeigneter Stelle im Freien. Seine Zieltemperatur regelt ein Thermostat, der mit einem NTC-Widerstand 10 kOhm / -200 Ohm/K die Temperatur mißt
- Der Sensor wird von einem USB-Netzteil (stabilisiert 5 Volt / 1 Ampere) versorgt
- Eine Roboremo-App gestattet den Dialog unter Android. Geheizt wird mit einem 8 Ohm / 3 Watt Widerstand, der von einem Power-MOS-Transistor (n-Chl.) geschaltet wird
- Eine blaue LED signalisiert den Bluetooth-Verbindungsstatus:
  - Schnelles Blinken: nicht verbunden
  - Dauernd an: verbunden
- Mit einer Ein/Aus-Taste kann die Heizung des Sensors abgeschaltet werden (Standby-Mode)
- Die nichtlineare Kennlinie des NTC wurde vorab mit einem Scilab-Script berechnet. Vier Referenzpunkte wurden gemessen, um daraus ein Korrekturpolynom zu bestimmen

Konzept

- Der Sensor arbeitet mit einem Arduino-pro-Mini auf Basis des Prozessors Atmel ATmega328, der über einen FTDI232-Adapter programmiert wird
- Angeschlossen über eine Software-UART ist ein Bluetooth-Modul CZ-HC-05, dessen 5-Volt-Kompatibilität und dessen einfache Verbindbarkeit mit Android Mobiltelefonen von Vorteil ist
- Der PowerMOS-Transistor der Heizung braucht eine möglichst hohe Gatespannung, um verlustfrei einzuschalten: hierfür sind 5 Volt günstig
- Der Sensor arbeitet als Thermostat, d.h. er hält die einmal eingestellte Zieltemperatur aufrecht
- Diese wird im EEPROM gespeichert, und bei Neustart geladen
- Zur Temperaturermittlung dient ein NTC-Widerstand von 10 kOhm mit einem negativen TK von etwa -200 Ohm pro Kelvin (NTC: Negative Temperature Coefficient)
- Da NTC eine schwerwiegend nichtlineare Kennlinie besitzen, wurde diese vorab mit dem ADC des ATmega vermessen
- Über ein Scilab-Programm wurde eine Ausgleichsparabel zur Linearisierung bestimmt, die als Array im *.ino File auftaucht

Schaltplan



Der FTDI232 wird nur zum Programmieren gesteckt

U2
CZ_HC-05
LED
KEY
RX
TX
GND
VCC

U1
Arduino_Mini
D9 D8 D7 D6 D5 D4 D3 D2 GND /RST RX TX
A4_SDA
A5_SCL
GND
A6
A7
DTR
TX
RX
VCC
GND
BLK
D10 D11 D12 D13 A0 A1 A2 A3 VCC /RST GND RAW

M1
DTR
RX
TX
5V
GND
GND
USB-mini
FTDI232_UART

R2
1k8
LED1
blue

VCC

R3
68k
NTC1
HL10k

D1
1N5819
C1
470µ 10V

BU1
5V / 1A
stab.
HEF_5.5_2.1

R1
8Ohm/3Watt
T1
IRF3205

Bluetooth-Modul
CZ-HC-05
Verbinden:
"HC-05" "1234"
9600 Baud

Materialien

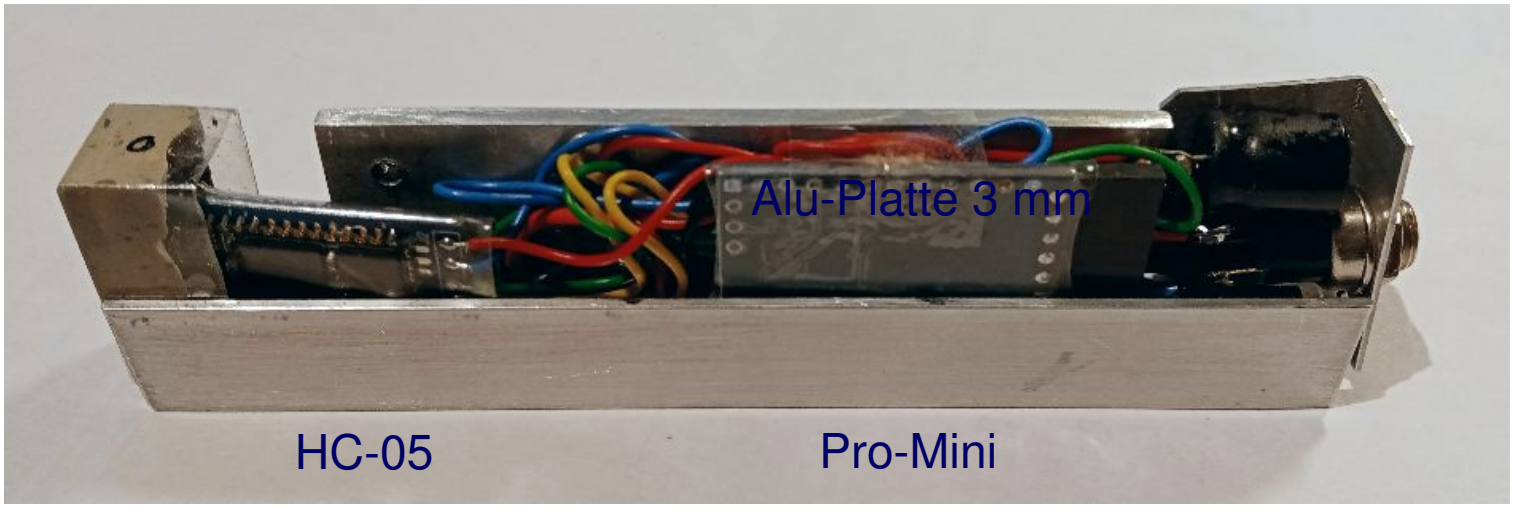
- Arduino-pro-Mini Board (5 Volt / 16 MHz) mit ATmega168 oder ATmega328
- Bluetooth-Modul CZ-HC-05 (BT-Modul mit VCC 5V!)
- Programmer FTDI232
- Power-MOS Transistor IRF3205 (55V / 75A) o.ä. 5V am Gate → >1A Ids
- Heizwiderstand 8 Ohm / 3 W, damit sind maximal 3 Watt Heizleistung möglich $P = U^2/R = 25\ V^2/\ 8\ Ohm = 3{,}125\ W$
- NTC-Widerstand 10 kOhm (24°C); TK = -200hm / Kelvin, Speisung über R3
- Blaue LED als Zustandsanzeige für BT-connection
- Kleinteile siehe Schaltplan

Bemerkungen zur Schaltung

- Die Schottky-Diode D1 dient der Entkopplung von Heizung und USB – wichtig, damit die Heizung nicht die USB-VCC belastet
- Der Bluetooth-Modul ist an Pins 2-3 angeschlossen, damit sich Bluetooth-Kommunikation und Hochladen nicht gegenseitig stören können (gefährlich)

Gehäuse

- Wichtig ist eine gute Wärmeverteilung im Aluminium Gehäuse
- Als Gehäuseschalen wurden zwei U-Profilstücke 15,5 x 1,5 mm; 125 mm lang zugeschnitten
- Diese werden mit einer 3 mm dicken Alu-Platte 27 x 85 mm verbunden
- Dazu werden in die Aluplatte vier Gewindebohrungen M3 gesetzt, Schrauben auf der Rückseite
- Heizwiderstand R1, Transistor T1 und NTC1 sind auf der Alu-Platte mit schraubbaren Klemmen tief unten versenkt
- Pro-Mini-Platine und Bluetooth-Platine sind mit Schrumpfschlauch isoliert



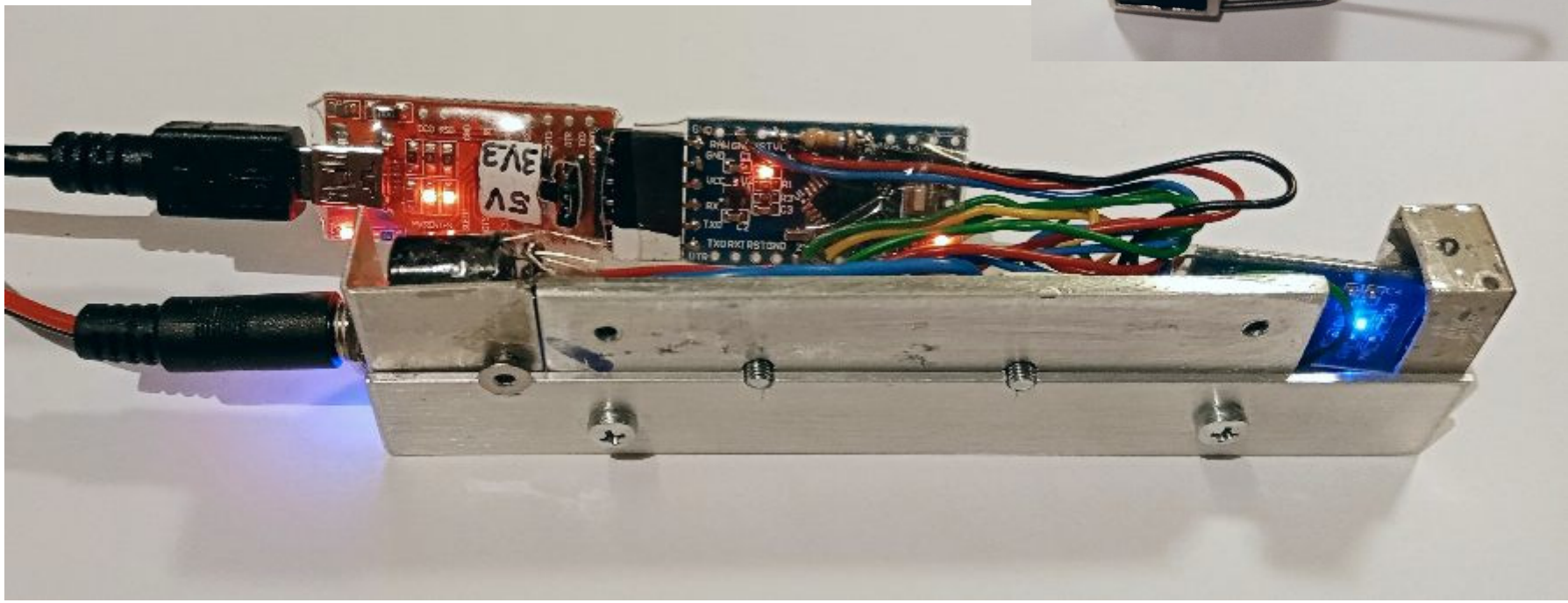
Rückseite

- Beide U-Profile sind hinten an der Alu-Platte verschraubt
- Die Antenne des HC-05-Moduls schaut aus dem "Dach" heraus
- Das "Dach" wurde aus Epoxy-Knetmasse geformt

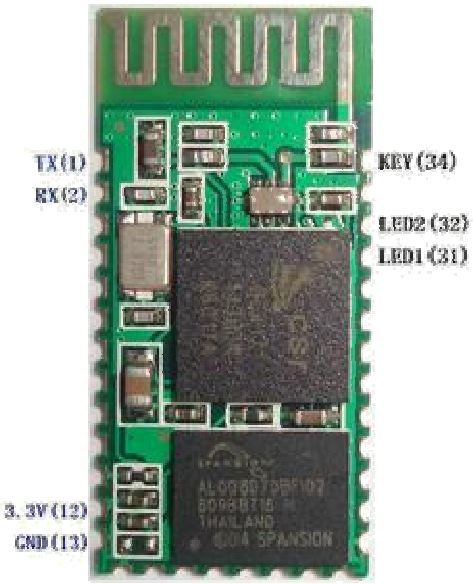
HC-05-Modul

Beim Programmieren

- Hier ist gerade der FTDI-Programmer gesteckt, um auf den Mini hochzuladen
- Man erkennt die dicke Aluplatte
- Rechts ist der HC-05 Modul zu erkennen



Bluetooth-Modul

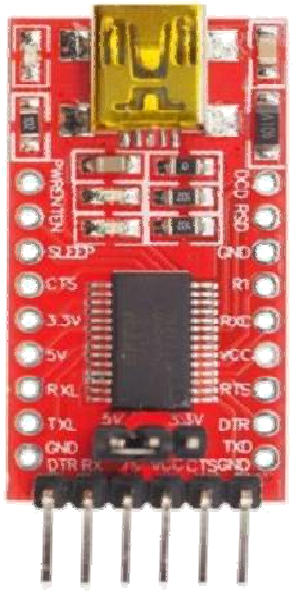


Verbindung Arduin-pro-Mini zum Bluetooth-Modul:

CZ-HC-05	Arduino	
GND	GND	
VCC	VCC	
TX-O	2	// Soft-UART RX
RX-I	3	// Soft-UART TX
LED	7	// als Arduino-Input

- Der Modul CZ-HC-05 arbeitet mit 9600 Baud; 115200 Baud schafft er nicht
- Android Pairing mit Name "HC-05", voreingestelltes Paßwort "1234"
- Im Master-Mode erfolgt automatisches Pairing mit vorgegebenen Gerät, sobald VCC anliegt und Roboremo gestartet wird (default)
- Reichweite im Alu-Gehäuse 5…10 Meter
- Der verwendete CZ-HC-05 hat eine VCC von 5 Volt (als einer der wenigen) mit integriertem Wandler 5V nach 3.3V (Bild oben ohne Trägerplatine)

Arduino-Kompilation



Arduino-Compiler (v.1.8.9) einstellen auf

- Werkzeuge/Board: "Arduino Pro oder Pro Mini"
- Werkzeuge/Prozessor: "ATmega168 (5V, 16 MHz)"
- Port: "..." COM-Port eintragen
- Programmer: FTDI232 USB-UART, Bild rechts

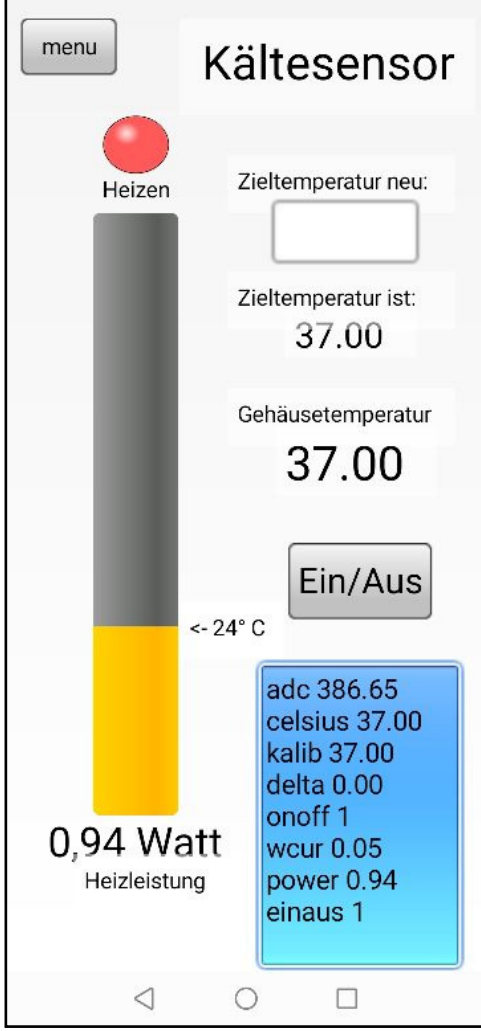
Achtung: Den Mini gibt es mit 5V/16MHz und mit 3.3V/8MHz

- Pro-Mini mit ATmega328 geht genauso – aber nur die 5-Volt-Version)
- Der Arduino-Nano kann ebenso benutzt werden, die Pin-Nummern dürften sogar identisch sein

Arduino-Compiler-Bug:

- Ausgaben sollten ursprünglich als "String"-Komposition erfolgen.
- Klappt nur bis zu fünf Ausgaben! Danach ominöse Compiler- und USB-Übertragungsfehler.

Android-GUI mit der Roboremo-App



Ausgaben (IDs) an USB und an Bluetooth für Roboremo:

```
adc 656        // ADC-Mittelwert
celsius 35.10  // "Gehäusetemperatur" in °C
kalib 35.00    // "Zieltemperatur ist" des Gehäuses
delta -0.10    // Ziel-Abweichung Thermostat in °C
onoff 0        // Ein-/Aus der Heizung - Roboremo-LED
wcur 0.05      // adaptive Wichtung der HMA
power 0.96     // mittlere Leistung in Watt
einaus 1       // Heizung dauernd abschalten = 0
```

Eingaben zum pro-Mini von der Roboremo-App:

```
kal 35         // "Zieltemperatur neu" Eingabe
einaus         // Heizung komplett abschalten
```

Einstellungen in Roboremo (1)

- Rot: Identifier zur Kommunikation mit dem Sensor
- Roboremo-Manual siehe roboremo.app

Vorsicht bei Roboremo:

- Lock edit: Das Schloß in der Mitte verhindert weiteres editieren!
- Rücksetzen: Menu -> Interface -> unlock edit all (see Manual, p.10) und alles von vorn.

Bedienelemente:

"button": Umschalten auf Standby (Heizung aus)

Text:	Ein/Aus
T.Size	24.0
Action:	remote
Press-ID:	einaus
Release-ID:	(freilassen)

Einstellungen in Roboremo (2)

"LED": Thermostat schaltet Heizung on/off

ID:	onoff	
Label:	On/Off	
ON cmd:	1	
OFF cmd:	0	
Color:	R	// für Red

"kbd connector": Eingabe des ADC-Zielwertes für Zieltemperatur

ID:	kal
Label:	Zieltemperatur
Action:	remote
Send:	when press enter

Einstellungen in Roboremo (3)

"text field": Ausgabe des Zielwertes für Gehäusetemperatur

ID:	kal	

"text field": Ausgabe der erreichten Gehäusetemperatur

ID:	celsius	

"Level Indicator": Balkenanzeige, Ausgabe in Watt

ID:	power	// ID "power" ist die Referenz
Label:	#*1 Watt	// #*1 gibt den Wert aus
Dec.count:	2	// auszugebende Label-Stellen nach dem Komma
T.Size:	28.0	// Schriftgröße des Labels
min.	0	// Ausgabe Minimum
max.	3.0	// Ausgabe Maximum
Color:	Y	// Balkenfarbe Yellow

Einstellungen in Roboremo (4)

Lästiges Springen ins Menue verhindern:

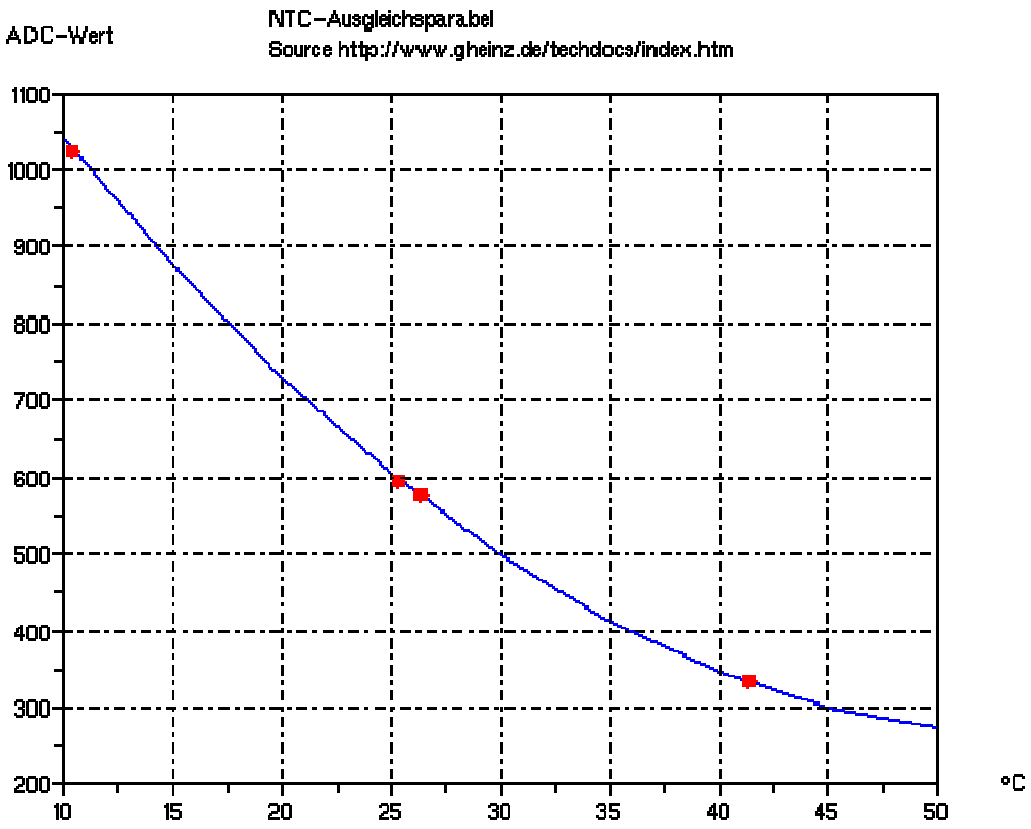
- Bei "Interface" → Touch: "exact" einstellen

Übertragen eines Roboremo-Interfaces
von einem Smartphone zum anderen über Bluetooth:

- *.interface mit "Export" speichern in Roboremo-Ordner (bei Import nachsehen, wo der Ordner zu finden ist)
- auf anderem Smartphone Roboremo installieren und nach dem Ordner schauen
- Kopieren auf anderes Smartphone über "Teilen" via Bluetooth
- in Roboremo-Ordner verschieben
- Interface: Roboremo: "Import" des Interfaces auf zweitem Smartphone

NTC-Linearisierung, Scilab-Ausgabe

- Scilab gibt ein Plot der Ausgleichsparabel aus
- Als Eingaben dienten die vier roten Wertepaare (Punkte)
- 11°C: in Garage gemessen
- 42°C: bestimmbar mit Fieberthermometer
- 25 und 26°C: im Zimmer meßbar



Achtung!

- Der Widerstand R3 (68k) bestimmt die tiefste, meßbare Temperatur
- 10-Bit ADC schlägt bei 1023 an <-> 1,1 Volt am ADC-Eingang

Linearisierung des NTC

- Im Datenblatt eines NTC ist i.a. ein Linearisierungspolynom zu finden
- Das wäre ein enormer Rechenaufwand für jede Temperaturmessung
- Da wir den NTC-Wert mit dem ADC des ATmega328 messen, kommen dessen Ungenauigkeiten sowie sein TK hinzu

Bessere Lösung:

- Der Sensor wird auf Zieltemperatur temperiert. Wir lesen den ADC-Wert ab
- Auf diese Weise gelangen vier Referenzmessungen, die zur Bestimmung einer Ausgleichsparabel mit Scilab genutzt werden konnten
- Die Parabel wird mit neun Stützstellen ausgegeben und im Arduino als Array eingelesen
- Nun braucht der Arduino nur noch das Intervall zu bestimmen, in welchem der gemessene ADC-Wert liegt
- In diesem Intervall wird die genaue Temperatur mit der scale() Funktion interpoliert

Viel Erfolg beim kreativen Nachbau!